Dynamic Load Balancing Model: Preliminary Assessment of a Biological Model for a Pseudo-Search Engine

Reginald L. Walker

Computer Science Department
University of California at Los Angeles
Los Angeles, California 90095-1596
rwalker@cs.ucla.edu

Abstract. Emulation of the current World Wide Web (WWW) search engines using methodologies derived from Genetic Programming (GP) and Knowledge Discovery in Databases (KDD) were used for the Pseudo-Search Engine's initial parallel implementation of an indexer simulator. The indexer was implemented to follow some of the characteristics currently implemented by AltaVista and Inktomi search engines who index each word in a Web document. This approach has provided very thorough and comprehensive search engine results that have led to the development of a Pseudo-Search Engine Indexer which has in turn provided insight into the computational effort needed to develop and implement an integrated search engine - information crucial to the adaptation of a biological model. The initial implementation of the Pseudo-Search Engine Indexer simulator used the Message Passing Interface (MPI) on a network of SUN workstations and an IBM SP2 computer system.

1 Introduction

Improvements to the fitness measure associated with Genetic Programming (GP) applications have taken various approaches. The evolution of species within a

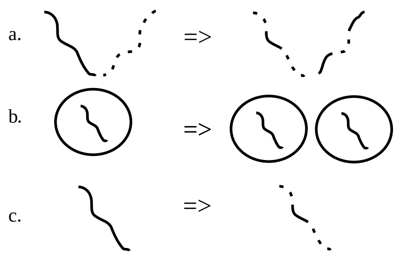

Fig. 1. Genetic programming operators a) crossover, b) reproduction, and c) mutation.

GP/Genetic Algorithms (GAs) search space results from an application of cluster analysis techniques that utilize the evaluation of each individual's (of a population) fitness measure. The search space associated with GP/GAs applications is obtained via a re-combination of partially-fit individuals who form potentially fitter offsprings. Thus, the re-combination operators provides the algorithm with a means to intelligently search [1] the entire search space with greater rewards. The similarity measures [8] used to compare and/or cluster individuals within the population are derived from the dimensionless ratio of the distance between two individuals and their maximum possible distance which is the search space's boundaries.

The use of a single population leads to panmictic selection [5] in which the individuals selected to participate in a genetic operation can be from anywhere in the population. The use of subpopulations (evolution of species) is an additional method used to avoid local optima [9]. Also, the GP operators adhere to the closure property [1],[9] because the primary and secondary operators generate a sets of functions and terminals that provide input to other applications of the genetic operators.

2 Methodologies of Genetic Programming

2.1 Overview

Genetic Programming is an evolutionary methodology [2],[4],[14] that extends and expands upon the techniques associated with Genetic Algorithms. The evolutionary force of these methodologies reflects a population's fitness. The foundation for GAs is an artificial chromosome of a fixed length and size designed to map the points in the problem search space. The artificial chromosome is derived by assigning the variables of the problem to specific locations (genes/alleles). The gene value denotes the value of a particular set of gene variables (memes) [1]. Such GAs provide an efficient mechanism for generating/displaying multidimensional search spaces that are highly complex and/or nonlinear.

The hybrid chromosome structure associated with the Pseudo-Search Engine's indexer [13] as shown in Figure 2, follows the methodologies of GP and GAs. Here, a single structure was used to represent subsets (subpopulations) of Web pages that reside at each node (Web site) and that will be eventually be expanded to an allocation of two chromosomes per node. In the chromosome each horizontal member structure represents a Web page that would translate into a meme - genetic component that varies with each allele. The bracket to the left of the Web pages refers to the pages having similar characteristics that comprise a allele and its memes - which represents/corresponds to the most fundamental features contained at each Web site. The addition of new Web pages at a given Web site creates new allele which can in turn grow in size via the addition of new memes (a process that does not alter the chromosome's length). The application of GP crossover operator - which transmits and contains the parents' genetic makeup - results in two new chromosomes. The bracket mechanism here

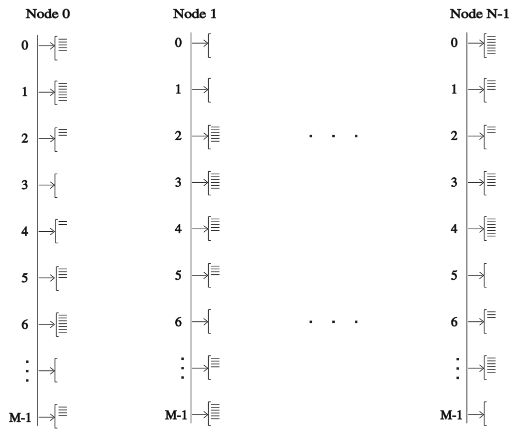

Fig. 2. Distribution of Web Pages.

constitutes an organizational device that numerically orders the structure's Web pages - a phenomenon that facilitates the evolution of diverse nodes.

2.2 Application of the Genetic Operators

The improvement of the solutions generated in the GP applications [9] is due to their evolution over a series of generations. Each successive generation is derived from an application of some combination of the primary genetic operators: *reproduction* and *crossover* (see Figure 1). And while the application of the former results in copying an individual into a new generation and/or subpopulation, the utilization of the latter results in a random selection of distinct points from two selected parents. Based on the values indicated by the fitness measure for two offspring - two potential members of a new generation - certain individuals of the population are stochastically selected as parents. Thus, the resulting children replace their parents in the existing population of solutions associated with a specific GAs/GP application. The migration operator in GP purges an existing sub-population of the least desirable members and in some cases the best member. This process provides GP with another mechanism to avoid local optimals.

Secondary operators, such operators are the *mutation* and *editing* operators, utilize a probability model that modifies individuals prior to their inclusion/addition to the new generation. By randomly selecting an individual and replacing a sub-tree with a new, randomly generated sub-tree the mutation operator avoids local optima in the search space. The editing operator in turn, chooses sets of individuals only when they are in possession of some specified generational frequency. After the application of the editing operator, the identities of the individuals are modified with simpler statements

Table 1. Genetic Programming and its corresponding Evolutionary Operators.

cause	Honeybee Colony	Genetic Programming Operators
bees fly out	migration/swarming	migration
queen and drones mate	reproduction	crossover/reproduction
worker bee lays eggs	mutation	mutation/editing

3 Adapting the Biological Model

3.1 Overview of the Biological Model

The basic characteristics exhibited by the inhabitants of a honeybee colony [3][12] are those that will be utilized to adequately search the Web for valuable information. These basic characteristics will be translated so as to develop a model of an integrated search engine using GP (see Table 1). The evolutionary processes exhibited by the bee colony thus provide a biological model not only for the storage, processing, and retrieval of valuable information but also for Web crawlers, as well as for an advanced communication system.

The biological storage mechanism for Web documents will simulate the interior components of the honeycomb, while the crawler mechanisms will be simulators that emulate the behavior of forager bees (communicating agents). Since the queen is the centralized control and co-ordination mechanism in a bee colony, queen simulators will be developed to control and coordinate all the process interactions within the simulated hives (Web sites). For mating purposes drone simulators will also be introduced in the model. Some of the functionality of the worker bees are integrated as components of a traditional operating system. Worker bees perform various tasks such as: 1) cell cleaning, 2) feeding larvae, 3) comb building, 4) nectar reception within the hive, 5) pollen packing, 6) removing debris, and 7) guarding. Also, they perform routine inspections and give extensive care to the bee larvae that range from one to thousands of visits. Current operating systems (OS) mechansims perform such rudimentary worker bees tasks as the allocation/deallocation of memory, garbage collection, and the basic form of computer security for user data as well as the OS itself.

The characteristics provided by this model remove the implementation limitations inherent in a methodology when developing a comprehensible model. The worker bees' conversion of pollen and nectar into honey will be simulated so as to provide a model for the indexing mechanisms that are essential in the organization of various sets of existing Web documents.

The current Pseudo-Search Engine Indexer, capable of organizing limited subsets of Web documents, provides a foundation for the first bee hive simulators. Adaptation of the honeybee model for the refinement of the Pseudo-Search Engine establishes order in the inherent interactions between the indexer, crawler and browser mechanisms by including the social (hierarchical) structure and simulated behavior of this complex system [6]. The simulation of behavior will engender mechanisms that are controlled and co-ordinated in their various levels of complexity.

3.2 Description of the Computer Model

The components of the computer system for the Pseudo-Search Engine can be compared to those of the honeybee colonies' social organization. The model in Table 2 shows the correlation between the honeycomb cells, queens, worker bees, drones, forager bees, guard bees, and such components as propolis, water, pollen, and nectar that are needed to sustain life in the colony. Here, the honeycomb cells, used to store pollen, nectar, and honey as well as brood (eggs and larvae), correspond to the memory cells in a computer. Each queen, responsible for controlling and coordinating most of the activities within a bee colony by releasing over 32 distinct pheromones, is here equivalent to a Web site (each indexer incorporates the functionality of the computer operating system) which manages and coordinates all of the activities associated with the efficient operation of a computer system. The implementation of drone Web sites will also be modeled after that of queen Web sites. Since the forager bees while searching for propolis, pollen, nectar, and sometimes water still serve the queen, the pheromones also regulate the activities that occur around the bee colony.

Most processes associated with the computer model for the Pseudo-Search Engine will have short life spans that are similar to those of the bee colony members. When submitted via the browser interface, the validity of a users requests will be determined by the guard bee emulators that inspect all the bees entering the colony. The forager bees here represent the Web crawlers which will retrieve Web pages from a diverse set of locations; and the diverse quantity of pollen, nectar, honey, and possible water that resides in a bee colony will here represent the diverse set of Web pages that comprised the WWW.

Table 2. Computer Model for the Evolutionary Pseudo-Search Engine.

Honeybee Colony	Computer Model
queens and drones	Web sites
propolis, pollen, nectar, and sometimes water	Web pages
honey	useful/organized information resulting from processed Web pages
honey comb cells	memory locations
brood and worker bees	processes
guard bees	computer security system
possible intruder - checked by guard bees - restricted internal access	possible user - restricted internal access

4 Computation Measures for the Pseudo-Search Engine

4.1 Overview

Computation measures [5] for GP applications were developed to measure the extent of the difficulty associated with the size of the population. The effort needed to apply genetic operators such as reproduction, crossover, and mutation is minute when compared to the computational effort needed to evaluate the fitness of each individual in a population. Also, as the population size increases, memory constraints become the next factor that requires consideration.

The creation of subpopulations requires the computation of localized fitness measures - a process that improves the overall performance to a degree that makes it directly proportional to the number of nodes (Web sites). One side effect of using GP and/or GAs-based approach is the disorderly sequence of conditions and other operations associated with individuals in the population. The model under development consists of a network of workstations and a network of single-board computer (such as the IBM SP2). The limitations of such system configurations include ongoing management and maintenance of the independent processes on an unruly collection of machines. The configuration for this type of environment consists of a host computer that acts as the file server, a Program Manager, and the network of processing nodes (queens and drones).

4.2 Computational Measures

By incorporating several different computational measures [5] the performance of the genetically enhanced Information Retrieval (IR) system for the Pseudo-Search Engine Indexer (the simulator of the internals of the bee colony's internal mechanisms) can be measured. The components of these measure include 1) the cumulative probability of success by generation i, $P(M,i)$, 2) the population size, M, and 3) a target probability, z.

The number of fitness evaluation for the sequential version with panmictic selection is

$$x = M(i+1), \tag{1}$$

while the number of fitness evaluation for the parallel version is

$$x = \sum Q_d(i(d)+1). \tag{2}$$

Here, i represents the generation number for the solution, d is the summation index that runs over the n nodes, Q_d is the subpopulation (deme) size, and $i(d)$ is the number of the last reporting generation from node d at the time when the sequential version satisfied the success criterion. The cumulative probability is computed via the following equation:

$$P(M,i) = \frac{\sum_i \ successful \ runs}{total \ number \ of \ runs}. \tag{3}$$

The number of independent runs needed to achieve a required probability, z, following x fitness evaluations is given by

$$R(z) = \left\lceil \frac{log(1-z)}{log(1-P(M,i))} \right\rceil . \tag{4}$$

The problem size [7] can be computed by using the equation

$$k = p \times g \times e, \tag{5}$$

where p is the population size, g is the number of generations, and e is the number of fitness cases. The number of Web pages to be classified is represented by the number of fitness cases for the Pseudo-Search Engine.

Additional fitness computations are associated with the maintenance of individuals of different species. The number of individuals [10] needed to produce a solution by generation i with probability greater than $z \approx 99\%$ is

$$I(M,i,z) = x \times (i+1) \times \left\lceil \frac{log(1-z)}{log(1-P(M,i))} \right\rceil . \tag{6}$$

This computation also indicates the number of individuals needed for the partial control mechanism [11] - a mechanism essential in the determination of the number of required Web crawlers. Another computational measure [5] used to determine the "computational effort" required in solving the given genetic programming system is computed via the following equation:

$$computational\ effort \;=\; min(I(M,i,z)). \tag{7}$$

5 Conclusion

In supplying its IR system current search engines incorporate some form of a crawler mechanism for the retrieval of Web documents. The implementation issues associated with the Pseudo-Search Engines' indexer provide insight into the caliber of adaptive crawler mechanisms essential in accurately parsing, indexing, and retrieving Web documents. These insights gained from the initial implementation of this simulator have been utilized in the initial phase of the biological model adaption. To facilitate communication among the diverse process mechanisms (the indexer(s), Web crawler(s), and the browser interface(s)), components indispensable in the formation of a fully integrated Pseudo-Search Engine, this biological model provides built-in mechanisms. In addition to representing a benchmark in the determination of the implemented scheme's efficiency, the adopted model also serves as a foundation for future evolutionary expansions of this search engine as World Wide Web documents continue to proliferate.

6 Acknowledgements

The author wishes to thank Walter Karplus and Zhen-Su She for their direction and suggestions. Support for this work came from the Raytheon Fellowship Program. Special thanks to Martha Lovette for her assistance with the figure in this paper.

References

1. Abramson, M.Z., Hunter, L.: Classification using Cultural Co-evolution and Genetic Programming. In: Koza, J.R., Goldberg, D.E., Fogel, D.B., Riolo, R.L. (eds.): Proc. of the 1996 Genetic Programming Conf. MIT Press, Cambridge, MA (1996) 249–254
2. Chapman, C.D., Jakiela, M.J.: Genetic Algorithm-Based Structural Topology Design with Compliance and Topology Simplification Considerations. J. of Mech. Design **118** (1996) 89–98
3. Free,J.B.: The Social Organization of Honeybees (Studies in Biology no. 81). The Camelot Press Ltd, Southampton (1970)
4. Koza, J.R.: Survey of Genetic Algorithms and Genetic Programming. In: Proc. of WESCON '95. IEEE Press, New York (1995) 589–594
5. Koza, J.R., Andre, D.: Parallel Genetic Programming on a Network of Transputers. Technical Report STAN-CS-TR-95-1542. Stanford University, Department of Computer Science, Palo Alto (1995)
6. Marenbach, P., Bettenhausen, K.D., Freyer, S., U., Rettenmaier, H.: Data-Driven Structured Modeling of a Biotechnological Fed-Batch Fermentation by Means of Genetic Programming. J. of Systems and Control Engineering **211 no. I5** (1997) 325–332
7. Oussaidène, M., Chopard, B., Pictet O.V., Tomassini, M.: Parallel Genetic Programming: An Application to Trading Models Evolution. In: Koza, J.R., Goldberg, D.E., Fogel, D.B., Riolo, R.L. (eds.): Proc. of the 1996 Genetic Programming Conf. MIT Press, Cambridge, MA (1996) 357–362
8. Senin, N., Wallace, D.R., Borland, N.: Object-based Design Modeling and Optimization with Genetic Algorithms. In: Banshaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith,R.E. (eds.): GECCO-99: Proc. of the Genetic and Evolutionary Computation Conf. Morgan Kaufman Publishers, Inc., San Francisco (1999) 1715–1721
9. Sherrah, J., Bogner, R.E., Bouzerdoum, B.: Automatic Selection of Features for Classification using Genetic Programming. In:Narasimhan, V.L. Jain, L.C. (eds.): Proc. of the 1996 Australian New Zealand Conf. on Intelligent Information Systems. IEEE Press, New York (1996) 284–287
10. Spector, L., Luke, S.: Cultural Transmission of Information in Genetic Programming. In: Koza, J.R., Goldberg, D.E., Fogel, D.B., Riolo, R.L. (eds.): Proc. of the 1996 Genetic Programming Conf. MIT Press, Cambridge, MA (1996) 209–214
11. Sinclair,M.C. , Shami, S.H.: Evolving Simple Software Agents: Comparing Genetic Algorithm and Genetic Programming Performance. In: Proc. of the 2nd Intl. Conf. on Genetic Algorithms in Engineering Systems: Innovations and Applications. IEE Press, London (1997) 421–426
12. von Frisch, K.: Bees: Their Vision, Chemical Senses, and Languages. Cornell University Press, Ithaca, New York (1964)
13. Walker, R.L.: Implementation Issues for a Parallel Pseudo-Search Engine Indexer using MPI and Genetic Programming. In: Proc. of the Sixth International Conf. on Applications of High-Performance Computers in Engineering. WIT Press, Ashurst, Southampton, UK (January 2000). To appear
14. Willis, M.J., Hiden, H.G., Marenbach, P., McKay, B. Montague, G.A.: Genetic Programming: An Introduction and Survey of Applications. In: Proc. of the 2nd Int. Conf. on Genetic Algorithms in Engineering Systems: Innovations and Applications. IEE Press, London (1997) 314–319